

AsbruView: Visualization of Time-Oriented, Skeletal Plans

Silvia Miksch and Robert Kosara

Vienna University of Technology, Institute of Software Technology
Resselgasse 3/188, A-1040 Vienna, Austria, Europe
silvia@ifs.tuwien.ac.at, rkosara@wvnet.at

Yuval Shahar and Peter Johnson

Section on Medical Informatics, Medical School Office Building, x215
Stanford University, Stanford, CA 94 305 - 5479, USA
shahar@smi.stanford.edu, pete@mimir.demon.co.uk

Abstract

Skeletal plans are a powerful way to reuse existing domain-specific procedural knowledge. The main drawbacks are that the compositions and the interdependencies of different skeletal plans and their components are not lucid. The aim of this paper is to overcome these limitations and to present the visualization of time-oriented, skeletal plans. Within the **Asgaard** project, we have developed a time-oriented and intention-based language, called **Asbru**, to represent such skeletal plans. The Asbru syntax is defined in Backus-Naur form (BNF). Reading BNF or similar forms are next to impossible even for domain experts. We explored different representations and automated knowledge-acquisition tools. However, the domain experts did not accept any of these representations. Consequently, we investigated different metaphor graphics and ended up with a plan visualization utilizing the metaphors of "tracks" and "traffic", called **AsbruView**. We formatively evaluated different approaches of this plan visualization with physicians applying treatment protocols of mechanical ventilated newborn infants.

Introduction and Motivation

Our approach is oriented, but not limited to our application domain: the medical (high-frequency) domain. Physicians are faced with two problems: (1) the information overload resulting from modern equipment, and (2) improving the quality of health care through increased awareness of proper disease management techniques. Treatment planning from scratch typically is not necessary, as general procedures exist which should guide the medical staff. These procedures are called *clinical guidelines* or *protocols*.

Appropriate clinical protocols are only available for a very limited class of clinical problems. Mostly, they are expressed in natural language, but this kind of representation can not easily be transformed into a formal and structured framework (Herbert 1994). Additionally, clinical protocols are not adjusted to the patient data-management system and they are partly vague and incomplete concerning their intentions and their temporal, context-dependent representation. Clinical protocols are a way of pre-compiling decisions that must be made, in which experts'

knowledge is distilled into a form of procedural knowledge. Extracting and formulating the knowledge structure for protocols is a non trivial task. Their implicit context must be made explicit. The variability of clinical protocols presents an additional challenge. A medical goal can be achieved by different therapeutic actions (e.g., in the domain of mechanical ventilation: pressure-controlled, volume-controlled, or ratio-controlled ventilation).

Besides using natural language, the most favored attempts to capture and support clinical protocols, are *flow diagrams* and *flowcharting tools*. Many medical experts are used to working with these techniques. Flow diagrams are useful for representing sequential states and actions in a graphical way. However, it is quite difficult to cope with all possible orders of plan execution and all the exception conditions that might arise. The trouble is that this by necessity can only cover a small subset of the possible situations and possible paths through. Additionally, flow diagrams are kinds of layering, which avoid to cope with concurrent (parallel) actions, with different temporal dimensions, with high numbers of possible transitions, and with mutual dependencies of parameters. A possible way to overcome these limitations are skeletal plans.

Similar problems were solved by procedural reasoning systems (PRSs, Georgeff, Lanskey, and Schoppers 1986) and situated and reactive planning (Firby 1989; Suchman 1987; Wilkins and Myers 1995). However, we need to have greater temporal reasoning power and focus on issues such as temporally extended goals (Bacchus and Kabanza 1996).

Skeletal Plans

A common strategy for the representation and the reuse of domain-specific procedural knowledge is the representation of that knowledge as a library of skeletal plans. Skeletal plans are plan schemata at various levels of detail that capture the essence of procedures, but leave room for execution-time flexibility in the achievement of particular goals (Friedland and Iwasaki 1985). Thus, they are usually reusable in different contexts.

A plan-specification language of skeletal plan needs to be expressive with respect to temporal annotations and needs to have a rich set of sequential, concurrent, and cyclical operators. Thus, it should enable designers to express complex procedures in a manner similar to a real

From: AIPS 1998 Proceedings. Copyright ' 1998, AAAI (www.aaai.org). All rights reserved.
programming language (although typically on a higher level of abstraction), but in a more appropriate and useful way. The language, however, also requires well-defined semantics for both the prescribed actions and the task-specific annotations, such as the plan's intentions and effects, and the preferences underlying them. Thus, the executing agent's actions can be better supported, leading to a more flexible dialog and, in the case of the clinical domains, to a better acceptance of automated systems for protocol-based care support. Finally, clear semantics for the task-specific knowledge roles also facilitate acquisition and maintenance of these protocols.

What Kind of Plan Visualization Do We Need?

We are aiming to support treatment planning using clinical protocols, which can be seen as skeletal plans. To apply skeletal plans to a dynamically changing environment, such as medicine, these skeletal plans have to capture all the requirements mentioned. The requirements in clinical domains are often a superset of the requirements in typical problem domains used in planning research.

We have developed a time-oriented, skeletal plan-specification language called **Asbru**. We tried to keep the language as simple as possible, however, we ended up with a quite complicated and difficult to comprehend language. The Asbru syntax is defined in Backus-Naur form (BNF). On one hand, reading BNF or similar forms are next to impossible for domain experts. On the other hand, flow diagrams, flow charts, or networks are not appropriate as mentioned above. An appropriate, different visualization of complicated plans is needed.

We need a plan visualization which is able to capture: (1) hierarchical decomposition of plans (which are uniformly represented in a plan-specification library); (2) time-oriented plans; (3) sequential, concurrent, and cyclical execution of plans; (4) continuous (durative) states, actions, and effects; (5) intentions considered as high-level goals; and (6) conditions, that need to hold at particular plan steps. Additionally, all different time-oriented components of skeletal plans should be visualized in an easy to understand way. The domain experts, such as physicians, should understand the basic idea of skeletal plans. However, the domain experts do not need to be familiar with the syntax of skeletal plans to author skeletal plans (clinical protocol).

Section 2 describes related techniques and their limitations. Section 3 gives an overview about the Asgaard project and the Asbru language. Section 4 sketches different approaches to describe Asbru using the medical scenario of mechanical ventilation. Finally, the main features of our plan visualization, called AsbruView will be explained.

Related Visualization and Knowledge-Acquisition Approaches

Visualization is concerned with exploring data and information in such a way as to gain understanding and new

insights into the data and the processes. The goal of visualization is to promote a deeper level of understanding of the data under investigation and to foster new insight into the underlying process (Tuft 1990; Tuft 1997).

In the last years, many visualization techniques were introduced to improve the understanding of the relationship between several variables (e.g., Cole and Stewart 1993; Frenkel 1988; Keller and Keller 1993). For example, Cole and Stewart (1993) suggested to use metaphor graphics (e.g., minute-ventilation rectangles representing the mechanical ventilator data) and found that the human performance to interpret mechanical ventilator data can be improved significantly (Cole and Stewart 1994).

Other approaches concentrated on the knowledge-based presentation of information, which plans how multimedial documents and procedures can be presented to various users (e.g., PPP and WIP (André 1997)).

Finally, there have been several efforts to create automated reactive planners to support the process and the acquisition of protocol-based care over periods of time (e.g., T-HELPER (Musen et al. 1992), PROMPT project (Fox, Johns, and Rahmanzadeh 1997)). On one hand, they are using automated (graphical) knowledge acquisition (KA) tools, such as PROTÉGÉ-II (Musen et al. 1995), to generate domain-specific knowledge-acquisition tools from ontologies. On the other hand, they are utilizing graphical workflow diagrams, similar to flow charts, to summarize and to assist in the correct and timely enactment of specific tasks, such as used in the PROMPT project (Fox, Johns, and Rahmanzadeh 1997).

The need of different methods for time-oriented, task-specific plan (and data) visualization—particular in modern intensive care units—is quite evident. An electronic version of flow charts and workflow diagrams are not applicable as discussed in Section 1. Plan-based presentation of information concentrates more on the user interactions and the static information presentation than on the visualization of the planning processes. Finally, the automated knowledge acquisition (KA) tools, do not support the time-oriented process- or plan-based concepts. They support acquiring static ontologies, however there is no assistance for the acquisition of time-oriented plans with different components.

The Asgaard/Asbru Project

The aim of the Asgaard/Asbru¹⁾ project (Shahar, Miksch, and Johnson 1996) is to design a planner based on time-oriented, skeletal plans. The planner will support the design and the execution of skeletal plans by a human executing agent other than the original plan designer. During design time the relevant tasks are: (1) plan verification and (2) plan validation. During execution time the relevant tasks are: (1) applicability of the plan to a particular state of the world, (2) guidance in proper execution of plans, (3) monitoring of

¹⁾ In Norse mythology, Asgaard was the home and citadel of the gods. It was located in the heavens and was accessible only over the rainbow bridge, called Asbru (or Bifrost).

the execution process, (4) assessment of the results of plans, (5) critiquing the execution process and its results, and (6) assistance in the modification of the original plan.

The underlying requirement to develop a task-specific planner is a plan-specification language. Therefore, within the Asgaard project, we developed a time-oriented, skeletal plan-specification language, called Asbru (Miksch, Shahar, and Johnson 1997). During the design phase of plans, Asbru provides a powerful mechanism to express durative actions and plans caused by durative states of an observed agent (e.g., many actions/plans need to be executed in parallel or periodically). These plans are combined with intentions of the executing agent of the plan. They are uniformly represented and organized in the *plan-specification library*. During the execution phase an applicable plan is instantiated with distinctive arguments (e.g., time-oriented patient data) and state-transition criteria (e.g., activated, completed, suspended, or aborted) are added to execute and to reason about different tasks. The intentions underlying these plans are represented explicitly as temporal patterns to be maintained, achieved or avoided.

A plan consists of a name, a set of arguments, including a time annotation (representing the temporal scope of the plan), and five components: **preferences**, **intentions**, **conditions**, **effects**, and a **plan body** which describes the actions to be executed. The general arguments, the time annotation, and all components are optional.

Preferences bias or constrain the selection of a plan to achieve a given goal and express a kind of behavior of the plan (e.g., implicit utility functions).

Intentions are high-level goals at various levels of the plan, an annotation specified by the designer, which supports tasks such as critiquing and modification. Intentions are temporal patterns of executing-agent actions and external-world states that should be maintained, achieved, or avoided. Intentions may consist of:

- (1) *Intermediate-state*: the state(s) that should be maintained, achieved, or avoided during the applicability of the plan (e.g., the blood-gas levels are slightly below to slightly above the target range);
- (2) *Intermediate-action*: the action(s) that should take place during the execution of the plan (e.g., minimize level of mechanical ventilation);
- (3) *Overall-state-pattern*: the overall pattern of states that should hold after finishing the plan (e.g., patient had less than one high blood-gas value per 30 minutes);
- (4) *Overall-action-pattern*: the overall pattern of actions that should hold after finishing the plan (e.g., avoid hand-bagging).

Conditions are temporal patterns, sampled at a specified frequency, that need to hold at particular plan steps to induce a particular state transition of the plan instance. We do not directly determine conditions that should hold during execution. We specify different conditions that enable transition from one plan state into another. A plan is completed when the completed conditions become true, otherwise the plan's execution

suspends or aborts. We distinguish between:

- (1) *Filter-preconditions* need to hold initially if the plan is applicable, but can not be achieved (e.g., subject is female). They are necessary for a plan to become possible;
- (2) *Setup-preconditions* need to be achieved to enable a plan to start (e.g., inspiratory oxygen concentration F_iO_2 is less than 80%) and allow a transition from a possible plan to a ready plan;
- (3) *Suspend-Conditions* determine when an activated plan has to be suspended—certain conditions (*protection intervals*) need to hold (e.g., blood gas has been above the target range for at least five minutes);
- (4) *Abort-Conditions* determine when an activated, suspended, or reactivated plan has to be aborted (e.g., the increase of the blood-gas level is too-fast for at least 30 seconds);
- (5) *Complete-conditions* determine when an activated or reactivated plan has to be completed successfully (e.g., returning to spontaneous breathing);
- (6) *Reactivate-Conditions* determine when a suspended plan has to be reactivated (e.g., blood gas level is back to normal or slightly increased).

Effects either describe the functional relationship between the plan arguments and measurable parameters or specify an overall effect of a plan. Effects have a likelihood annotation—a probability of occurrence.

The **plan body** is a set of plans to be executed in parallel, in sequence, in any order, or in some frequency. We distinguish among several types of plans: *sequential*, *concurrent*, and *cyclical*. Only one type of plan is allowed in a single plan body. A sequential plan specifies a set of plans that are executed in sequence; for continuation, all plans included have to be completed successfully. Concurrent plans can be executed in parallel or in any order. We distinguish two dimensions for classification of sequential or concurrent plans: the number of plans that should be completed to enable continuation and the order of plan execution. The continuation condition specifies the names of the plans that must be completed successfully to proceed with the next steps in the plan. A cyclical plan includes a plan that can be repeated, and optional temporal and continuation arguments that can specify its behavior.

A *plan* in the plan-specification library is composed hierarchically, using the Asbru syntax, of a set of plans with arguments and time annotations. A decomposition of a plan into its subplans is always attempted by the execution interpreter, unless the plan is not found in the plan-specification library, thus representing a nondecomposable plan (informally, an *action*). This can be viewed as a "*semantic*" *stop-condition*. Such a plan is referred to the agent for execution, which may result in an interaction with a user or an external calling of a program.

"Temporal Pattern" and "Time annotations"

Intentions, world states, and prescribed actions are temporal patterns. A temporal pattern is either a *parameter*

proposition—a parameter (or its abstraction), its value, a context, and a time annotation (e.g., the *state* abstraction of the blood-gas parameter is *normal*, as defined in the context of weaning therapy, during a certain time period)—, a *combination* of multiple parameter propositions, or a *plan-state* associated to an instantiated plan (plan pointer) and a time annotation.

The time annotations we use allows a representation of uncertainty in starting time, ending time, and duration (Dechter, Meiri, and Pearl 1991; Rit 1986). The time annotation supports multiple time lines (e.g., different zero-time points and time units) by providing *reference annotations*. Temporal shifts from the reference annotation are defined to represent the uncertainty in starting time, ending time, and duration, namely earliest starting shift (ESS), latest starting shift (LSS), earliest finishing shift (EFS), latest finishing shift (LFS), minimal duration (MinDu), and maximal duration (MaxDu). The temporal shifts are associated with time units (e.g., minutes, days). Thus, a temporal annotation is written as ([ESS, LSS], [EFS, LFS], [MinDu, MaxDu], REFERENCE). ESS, LSS, EFS, LFS, MinDu, and MaxDu can be "unknown" or "undefined" to allow incomplete time annotation, denoted by an underscore "_".

To allow temporal repetitions, sets of cyclical time points and cyclical time annotations are defined. Short-cuts are used to allow starting a plan immediately at the current time (using the symbol "*now*"), to use the activation of a plan as reference point (using the symbol "*self*"), or to allow that a condition holds during the span of time over which the plan is executed (using the symbol "*").

The Way to AsbruView: A Skeletal Plan Visualization

Having defined our Asbru language with all the different components, the next step was to evaluate its applicability with real clinical protocols. We tried various approaches to fill the generic skeletal plans with real clinical treatment protocols for infants' respiratory distress syndrome.

First, we will roughly explain the medical scenario in natural language. Second, we will show, how parts of the example look like in BNF-based Asbru syntax. Third, we will sketch our experiences using an automated KA tool and conclude with the main limitations of the former two approaches. These limitations lead to our metaphor graphics of "tracks" and "traffic", called AsbruView, which is illustrated afterwards. We performed scenario-based evaluations of the different approaches with physicians (Caroll 1995). The results of this evaluation influenced the final version of AsbruView.

Example: I-RDS (in natural language)

After infants' respiratory distress syndrome (I-RDS) is diagnosed, a plan dealing with limited monitoring possibilities is activated, called *initial-phase*. Depending on the severity of the disease, three different kinds of plans are followed: *controlled-ventilation*, *permissive-hypercapnia*,

or *crisis-management*. Only one plan at a time can be activated, however the order of execution and the activation frequency of the three different plans depend on the severity of the disease. Additionally, it is important to continue with the plan *weaning* only after a successful completion of the plan *controlled-ventilation*. After a successful execution of the plan *weaning*, the extubation should be initiated. The extubation can be either a single plan *extubation* or a sequential execution of the subplans *cpap* and *extubation*.

The most important part is the subplan *controlled-ventilation*. The intentions of this subplan are to maintain a normal level of the blood-gas values and the lowest level of mechanical ventilation (as defined in the context of controlled ventilation therapy) during the span of time over which the subplan is executed. This subplan is activated immediately, if peak inspiratory pressure PIP \leq 30 and the transcutaneously assessed blood-gas values are available for at least one minute after activating the last plan instance *initial-phase* (as reference point). The subplan must be aborted, if PIP $>$ 30 or the increase of the blood-gas level is too steep (as defined in the context of controlled ventilation-therapy) for at least 30 seconds. The sampling frequency of the abort condition is 10 seconds. The subplan is completed successfully, if $F_iO_2 \leq 50\%$, PIP ≤ 23 , $f \leq 60$, the patient is not dyspnoeic, and the level of blood gas is normal or above the normal range (as defined in the context of controlled ventilation-therapy) for at least three hours. The sampling frequency of the complete condition is 10 minutes. The body of the subplan *controlled-ventilation* consists of a sequential execution of the two subplans *one-of-increase-decrease-ventilation* and *observing*.

Example: I-RDS (in BNF-based Asbru Syntax)

```
(PLAN I-RDS-therapy ...
(DO-ALL-SEQUENTIALLY
  (initial-phase)
  (one-of-controlled-ventilation)
  (weaning)
  (one-of-cpap-extubation)))

(PPLAN one-of-controlled-ventilation ...
(DO-SOME-ANY-ORDER
  (controlled-ventilation)
  (permissive-hypercapnia)
  (crisis-management)
  CONTINUATION-CONDITION
    controlled-ventilation))

(PPLAN controlled-ventilation
(PREFERENCES (SELECT-METHOD BEST-FIT))
(INTENTION:INTERMEDIATE-STATE
  (MAINTAIN STATE(BG) NORMAL
    controlled-ventilation *))
(INTENTION:INTERMEDIATE-ACTION
  (MAINTAIN STATE(RESPIRATOR-SETTING) LOW
    controlled-ventilation *))
(SETUP-PRECONDITIONS
  (PIP (<= 30) I-RDS *now*)
  (BG available I-RDS
    [[_, _], [_, _], [1 MIN, _]
      (ACTIVATED initial-phase-l#)]))
(ACTIVATED-CONDITIONS AUTOMATIC)
(ABORT-CONDITIONS ACTIVATED
  (OR (PIP (> 30) controlled-ventilation
    [[_, _], [_, _], [30 SEC, _], *self*])
    (RATE(BG) TOO-STEEP controlled-ventilation
    [[_, _], [_, _], [30 SEC, _], *self*]))
  (SAMPLING-FREQUENCY 10 SEC))
```

```
(COMPLETE-CONDITIONS
(FiO2 (<= 50) controlled-ventilation
 [[_, _], [_, _], [180 MIN, _], *self*])
(PIP (<= 23) controlled-ventilation
 [[_, _], [_, _], [180 MIN, _], *self*])
(f (<= 60) controlled-ventilation
 [[_, _], [_, _], [180 MIN, _], *self*])
(STATE(patient) (NOT DYSPNOEIC)
 controlled-ventilation
 [[_, _], [_, _], [180 MIN, _], *self*])
(STATE(BG) (OR NORMAL ABOVE-NORMAL)
 controlled-ventilation
 [[_, _], [_, _], [180 MIN, _], *self*])
(SAMPLING-FREQUENCY 10 MIN))
(DO-ALL-SEQUENTIALLY
(one-of-increase-decrease-ventilation)
(observing)))
```

Example: I-RDS (using a KA-tool)

It is quite obvious that physicians will hardly use the Asbru plan-specification language to author clinical protocols. Therefore, we explored PROTÉGÉ-II/Win (the Windows version of PROTÉGÉ-II, Musen et al. 1995) to automatically generate a graphical KA tool. PROTÉGÉ-II is a set of tools and a methodology to develop knowledge-based systems. We used an object-oriented version of the Asbru language to develop the ontology. Once the ontology is defined, the PROTÉGÉ/Win LayoutEditor tool automatically generates a specification of a KA tool for this ontology. The specification of the KA tool is interpreted by the PROTÉGÉ/Win LayoutInterpreter. It is possible to change the layout of the user interface to some degree in the LayoutEditor. The resultant KA tool (see Figure 1) can then be used to acquire instances of the ontology, which in this case would be protocols in the Asbru language.

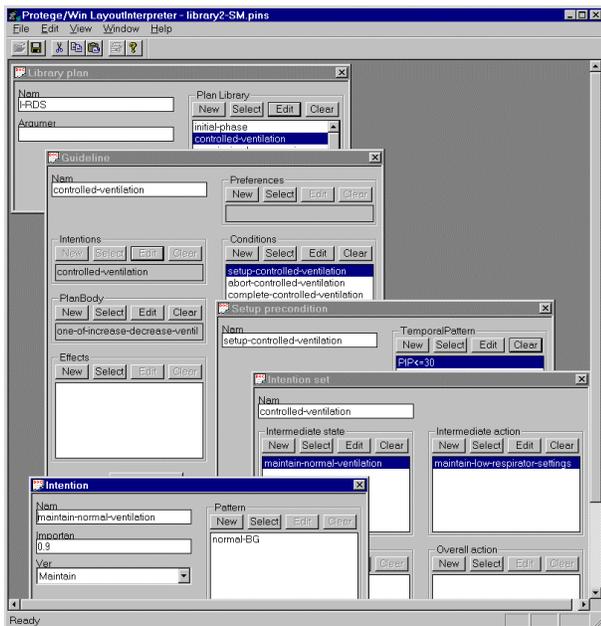


Figure 1: Screen shot of knowledge-acquisition tool, showing a part of I-RDS protocol.

Benefits and Limitations of BNF-based Asbru Syntax and the KA-tool

If the user, in particular a domain expert, is unfamiliar with the syntax, then it is easier to use the KA tool than the BNF-syntax. Another significant benefit of the KA tool approach is that it detects incorrect syntax while authoring a protocol. However, the complexity of the ontology enforces the automatic generator of the KA tool to produce a user interface with many cascading and small dialogs. ²⁾

Our collaborating physicians evaluated the different approaches. The physicians got easily lost within too many opened windows: they did not know, what to work on next, which parts of the protocol had already been filled with content and which parts were still missing, how the different skeletal plans and subplans were connected, how to read a plan in general, etc. These observations resulted in a poor acceptance and understanding of the Asbru language. Asbru's components are understandable to the language designers, however, they were not comprehensible to the medical staff. The physicians need more guidance (e.g, which parts of the plans are filled with knowledge), or a kind of simulation, how a possible way through the plan library could look like. Therefore, we were investigating in adding domain knowledge and visualization techniques, which resulted in our plan visualization utilizing the metaphors of "tracks" and "traffic", called AsbruView.

AsbruView

Our plan-visualization approach was influenced by the idea of metaphor graphics (Cole and Stewart 1993) and the graphical-timetable design of Shinkansen Lines (Japanese National Railroad) described in (Tufte 1990).

We have utilized metaphor graphics of "tracks" and "traffic" to envision different time-oriented skeletal plans. The basic element is a track, which represents a simple plan. Stacking techniques are used to visualize the decomposition of plans into subplans. We are using 3-dimensional objects. The width represents the time axis, the depth represents plans on the same level of decomposition, and the height represents the decomposition of plans into subplans. The cube is rotated to the left to enable plan labeling on the top of the track and to ensure readability in case of multiple tracks. Figure 2 shows the treatment protocol of I-RDS therapy, which is decomposed in four sequentially executed subplans: *initial-phase*, *one-of-controlled-ventilation*, *weaning*, and *one-of-cpap-extubation*.

We distinguish among several types of plans: *sequential*, *concurrent*, and *cyclical*. The sequential plan is shown in Figure 2. The additional four types of plans (parallel: DO-ALL-TOGETHER, DO-SOME-TOGETHER; in any order: DO-ALL-ANY-ORDER, DO-SOME-ANY-ORDER) are depicted in Figure 3. If plans should be executed in parallel means, they should start together, however, they do not

²⁾ We were told that additional features will be available for more control of the layout of the automatically generated user interface in PROTÉGÉ/Win soon.

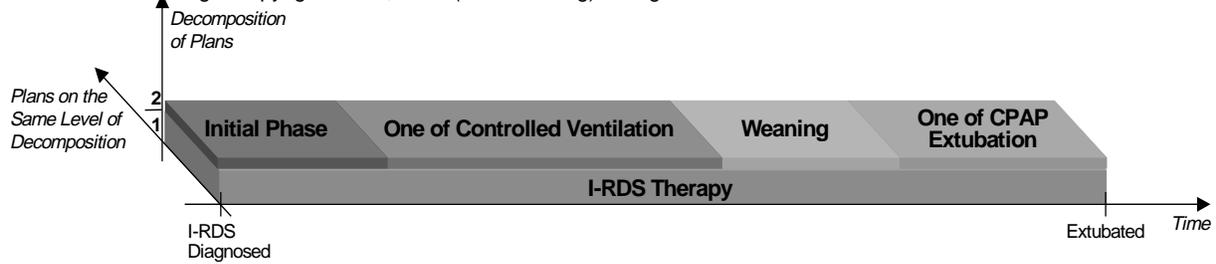


Figure 2: Stacking and sequential plans.

need to end at the same time. In case of "DO-SOME-TOGETHER", the dotted line represents the optional plans and the solid line represents the plans which must be completed successfully to proceed with the next steps in the plan. (called continuation condition). If plans are executed in any order, they may start and end at the same time, however it depends on their conditions and time annotations. In case of "DO-SOME-ANY-ORDER", the dotted arrows identify the optional plans and the solid arrows name the plans included in the continuation condition.

We are using different icons to visualize the six kinds of conditions (compare Figure 4). The sign "No Entry with Exceptions" symbolizes the filter-precondition. The supplementary sign stands for the exemptions, like "Except Busses", which we are using to name the filter-conditions (e.g., "Except Females" allows only females to enter the track). The setup precondition is embodied by a turnpike, which illustrates the fact that this condition can be achieved (and thus the turnpike will be opened). The traffic light includes three kinds of conditions: red light symbolizes the abort-condition, yellow light the suspend-condition, green light the reactivate-condition. The finishing-line (flag) stands for the complete-condition.

Our time annotation is illustrated in Figure 5. This kind of visualization is applied to the temporal patterns. The temporal dimensions of the plans and their subplans are given in the horizontal axis directly (compare Figure 6).

The general rule of *undefined components* is that the undefined icons appear in gray. For example, the undefined elements of the time annotation are gray in the lower part of Figure 5, or the red light of the traffic light is gray in Figure 6, which means that the abort-condition is missing.

To keep readability of the different components of the Asbru language, we use a control panel (see Figure 6). The control panel is used to choose which Asbru component should be visible (in Figure 6 the left-hand side of the control panel) and which levels of decomposition are visible (in Figure 6 the right-hand side of the control panel labeled "Visible Levels"). In Figure 6, the visible level informs that the first, second, and third levels of decomposition are visible (depicted by the arrows) and the conditions of the second level are shown (depicted by the marked check-box next to the label conditions and the big arrow next to the second visible level). The finishing-lines (complete-conditions) of all visible levels are shown to distinguish unambiguously between the different tracks.

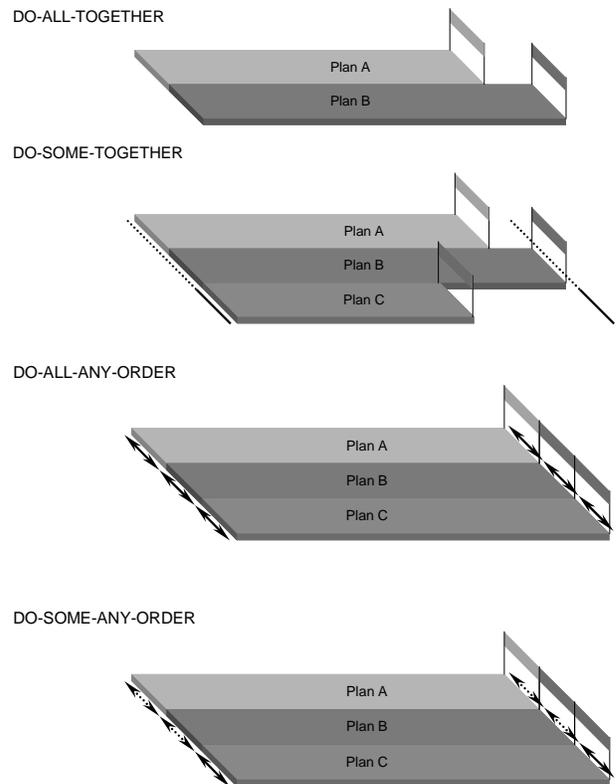


Figure 3: Plans to be executed in parallel (upper part) and in any order (lower part).

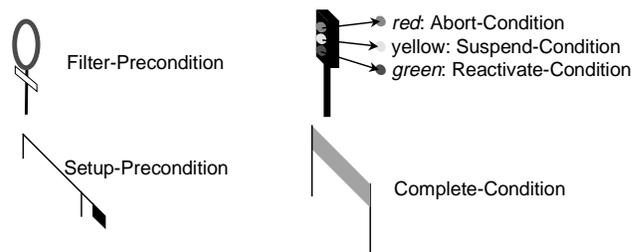


Figure 4: The icons of the six kinds of conditions.

Definition: [[ESS, LSS], [EFS, LFS], [MinDu, MaxDu], Reference]

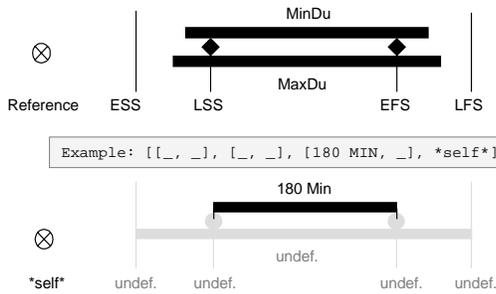


Figure 5: Asbru's Time Annotations: The upper part of the figure presents the generic annotation and the lower part shows an example.

Example: I-RDS (in AsbruView)

Figure 6 shows parts of the I-RDS protocol in AsbruView. Three levels of decomposition are visible. The big arrow in the control panel marks which conditions should be shown, namely the conditions of the second level. The conditions of the subplan *one-of-controlled-ventilation* are undefined, therefore all the icons (including the flag) are gray. The time annotation *"*self*", "+10 min"*, and the black triangle

means, the subplan *observe-blood-gas* should last for at most 10 minutes after the subplan itself is activated.

Figure 7 shows three levels of decomposition and the intentions of plan *controlled-ventilation* (the third level of decomposition is activated in the control panel). The time annotations of the two intentions are very simple, because a normal level of the blood-gas values and the lowest level of mechanical ventilation should be maintained during the span of time over which the subplan is executed.

Conclusion and Future Plans

We outlined the necessity for suitable plan visualization and showed the metaphor graphical approach AsbruView, which clarifies a complex plan-specification language in a comprehensible way. We have utilized the metaphors of "tracks" and "traffic". The applicability of AsbruView was evaluated with scenario-based techniques. We applied treatment protocols of mechanical ventilated newborn infants and analyzed AsbruView's expressiveness with collaborating physicians.

AsbruView is based on a time-oriented and intention-based language, called Asbru, to represent skeletal plans. Asbru places a particular emphasis on an expressive representation for time-oriented actions and world states in combination with the underlying intentions as temporal patterns

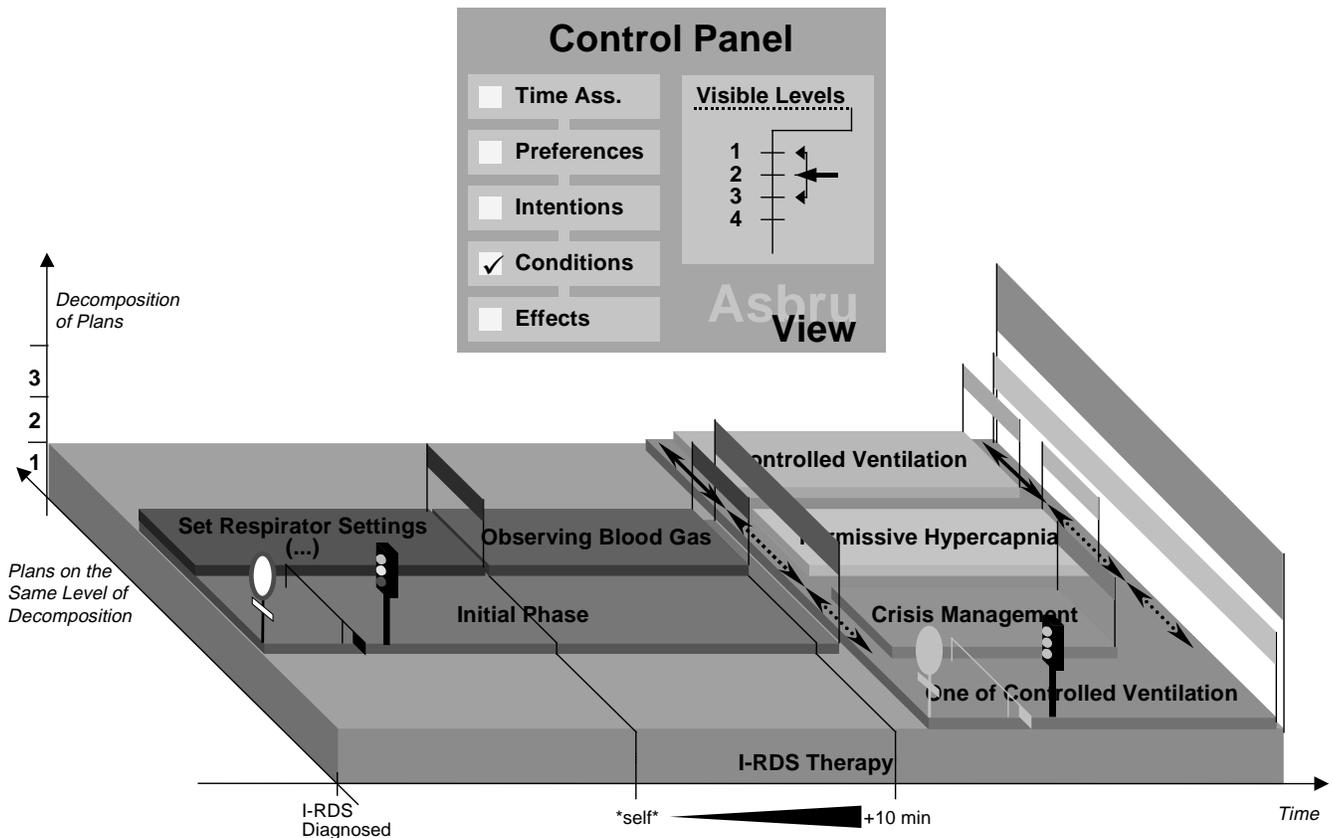


Figure 6: Parts of I-RDS treatment protocol in AsbruView: The conditions of subplans *initial-phase* and *one-of-controlled-ventilation* are activated (second visible level).

to be maintained, achieved or avoided. It allows to use different granularities and reference points to represent multiple time lines. Asbru's representation includes the duration of actions, their success or failure, and allows time annotation of events, actions/plans, and world states with uncertainty in their appearances. Asbru has a rich set of sequential, concurrent, or cyclical operators, which enable to express complex procedures. Preferences, intentions, conditions, effects, and actions are specified on various detail levels depending on their appearances and evidences. Such a complex plan-specification language is needed to capture all requirements of a dynamically changing environment, such as medicine.

AsbruView is able to visualize most of the features of Asbru in an easy to understand way and supports the navigation through a complex plan-specification library. Therefore, domain experts need not to be familiar with the Asbru syntax to author a plan.

Currently, we are implementing the AsbruView in Java. Additionally, we are improving the expressiveness of AsbruView concerning the time annotations of events, actions/plans, and world states with uncertainty in their appearances and the cyclical plan visualization. Our final aim is to use AsbruView during the design and the execution phase. Therefore, we will adapt AsbruView to be used to author a protocol during the design phase as well as to visualize the performed protocols during the execution phase in a user-appropriate and task-specific way.

Acknowledgments. The authors thank Johannes Gärtner, Werner Horn, Christian Popow, Franz Paky, Georg Duftscheid, and Klaus Hammermüller for helpful comments and discussions.

References

André, E. 1997. WIP and PPP: A Comparison of two Multimedia Presentation Systems in Terms of the Standard Reference Model Computer Standards and Interfaces. *Computer Standards and Interfaces*.

Bacchus, F. and Kabanza, F. 1996. Planning for Temporally Extended Goals, In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1215-1222. Menlo Park, CA: AAAI Press.

Caroll, J. M. 1995. *Scenario-Based Design - Envisioning Work and Technology in System Design*. New York: John Wiley&Sons.

Cole, W. G.; and Stewart, J. G. 1993. Metaphor Graphics to Support Integrated Decision Making with Respiratory Data. *Int. Journal of Clinical Monitoring and Computing*, 10:91-100.

Cole, W. G.; and Stewart, J. G. 1994. Human Performance Evaluation of a Metaphor Graphic Display for Respiratory Data. *Methods of Information in Medicine*, 33:390-396.

Dechter, R.; Meiri, L.; and Pearl, J. 1991. Temporal Constraint Networks. *Artificial Intelligence, Special Volume on Knowledge Representation*, 49(1-3):61-95.

Firby, R. J. 1989. Adaptive Execution in Complex Dynamic Worlds. Ph.D. diss., Yale University.

Fox, J.; Johns, N.; and Rahmanzadeh, A. 1997. Protocols for Medical Procedures and Therapies: A Provisional Description of the PROforma Language and Tools. 21-38. In *Proceedings of 6th Conference on Artificial Intelligence in Medicine Europe (AIME-97)*, Berlin: Springer.

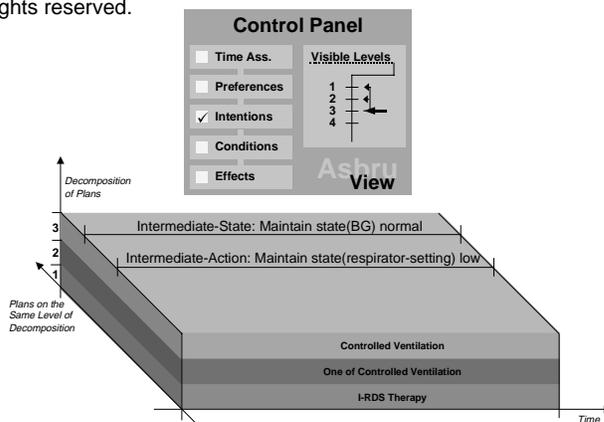


Figure 7: Intentions of plan controlled-ventilation.

Frenkel, K. A. 1988. The Art and Science of Visualizing Data. *Communications of the ACM*, 31(2):101-121.

Friedland, P. E.; and Iwasaki, Y. 1985. The Concept and Implementaion of Skeletal Plans. *Journal of Automated Reasoning*, 1(2):161-208.

Georgeff, M. P.; Lanskey, A. L.; and Schoppers, M. J. 1986. Reasoning and Planning in Dynamic Domains: A Experiment with Mobile Robots, SRI International, AI Center, TechNote 380.

Herbert, S. I. 1994. Informatics for Care Protocols and Guidelines: Towards a European Knowledge Model. In Gordon, C. J. and Christensen, J. P. eds. *Health Telematics for Clinical Guidelines and Protocols*, Amsterdam: IOS Press.

Keller, P. R.; and Keller, M. M. 1993. *Visual Cues, Practical Data Visualization*. New York: IEEE Press.

Miksch, S.; Shahar, Y.; and Johnson, P. 1997. Asbru: A Task-Specific, Intention-Based, and Time-Oriented Language for Representing Skeletal Plans. In *Proceedings of the 7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97)*, Milton Keynes, UK, Open University.

Musen, M. A.; Carlson, C. W.; Fagan, L. M.; Deresinski, S. C.; and Shortliffe, E. H. 1992. T-HELPER: Automated Support for Community-Based Clinical Research. In *Proceedings of the Sixteenth Annual Symposium on Computer Applications in Medical Care (SCAMC-92)*, 719-723. New York: McGraw Hill.

Musen, M. A.; Gennari, J. H.; Eriksson, H.; Tu, S. W.; and Puerta, A. R. 1995. PROTÉGÉ -II: A Computer Support for Development of Intelligent Systems from Libraries of Components. 766-770. In *Proceedings of the Eighth World Congress on Medical Informatics (MEDINFO-95)*.

Rit, J.-F. 1986. Propagating Temporal Constraints for Scheduling. 383-388. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, Los Altos : Morgan Kaufmann.

Shahar, Y.; Miksch, S.; and Johnson, P. 1996. A Task-Specific Ontology for Design and Execution of Time-Oriented Skeletal Plans. In *Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.

Suchman, L. A. 1987. *Plans and Situated Actions: The Problem of Human/Machine Communication*. Cambridge University Press.

Tufte, E. R. 1990. *Envisioning Information* Cheshire, CT: Graphics Press.

Tufte, E. R. 1997. *Visual Explanation* Cheshire, CT: Graphics Press.

Wilkins, D. E. and Myers, K. L. 1995. A Common Knowledge Representation for Plan Generation and Reactive Execution. *Journal of Logic and Computation*, 5(6):731-761.